Politechnika Rzeszowska

Wydział Budowy Maszyn i Lotnictwa

Katedra Mechaniki i Robotyki Stosowanej

Instrukcja do Laboratorium z KSP:
Port Szeregowy w Python+QT

Autor:
Paweł Penar

1 Cel ćwiczenia

Za pomocą języka Python, biblioteki QT oraz pakietu pySerial, zrealizować program okienkowy konfigurujący program *Serial Device Emulator*.

2 Wstęp do języka Python

2.1 Python - odnośniki do zasobów internetowych

Należy zapoznać się z następującymi stronami internetowymi

- `http://www.python.rk.edu.pl/w/p/podstawy/` Kurs Python'a
- `http://www.python.rk.edu.pl/w/p/pyqt/` Opis modułu PyQt
- `http://www.qt.io/developers/` Strona domowa projektu QT

Uwaga: Stosujemy **Python w wersji 2.7**

3 Pakiet *pyserial*

Pakiet *pyserial* oferuje API do Python'a które pozwala na łatwy dostęp do portu szeregowego w różnych systemach operacyjnych. Więcej informacji o pakiecie znajduje się na stronie domowej projektu, tj. `http://pyserial.sourceforge.net/`. Co więcej, **pakiet jest domyślnie zainstalowany w pakiecie *WinPython***. Dodatkowo, pod adresem `http://pyserial.sourceforge.net/shortintro.html`, znajduje się krótki wstęp do pakietu pokazujący API oraz podstawowe metody klasy *Serial*, która odpowiada za połączenie z portem COM.

Korzystanie z pakietu należy rozpocząć od importu pakietu do pliku skryptu, tj.:

```
import serial
```

Następnie do zmiennej *port*, przypiszemy nową instancje klasy *Serial*, która zostanie podłączona do portu *COM1*.

```
port=serial.Serial('COM1')
```

Dokładny opis klasy i dokumentacja konstruktora (wraz z domyślnymi parametrami połączenia z portem COM) znajduje się na stronie `http://pyserial.sourceforge.net/pyserial_api.html`

Poprawność połączenia z portem COM można sprawdzić za pomocą metody *isOpen*, np.:

```
print port.isOpen()
```

By wypisać nazwę portu, z którym został połączona instancja klasy *Serial*, zapisana pod zmienną *port*, należy wypisać pole *name* klasy *Serial*, tj.:

```
print port.name
```

Jeśli zaś chcemy wysłać ciąg liter na port szeregowy należy skorzystać z metody *write*

```
port.write('KSP')
```

Funkcją odwrotną do funkcji wysyłania, jest funkcja czytania, która występuje z różnymi parametrami. Najprostsza jej wersja jest bezparametrowa, a jej użycie jest następujące:

```
x=port.read()
```

W tym wypadku metoda *read* odczytuje jeden bajt danych, który może zostać wypisany przez *print*. Należy dodać, że metoda *read* blokuje program, dlatego często za jej wykonanie odpowiada osobny wątek.

Zamknięcie połączenia z portem COM jest realizowane przez metodę *close*

```
port.close()
```

Podsumowując, prosty skrypt używający pakietu *pyserial* prezentuje poniższy listing:

```
import serial

port = serial.Serial('COM7')
if port.isOpen():
    port.write('hello')
    x=port.read()
    print x
    port.close()
```

3.1 Przydatne uwagi

Poniżej znajdują się dwie uwagi dotyczące przesyłania za pomocą pakietu *pyserial* liczb całkowitych w zakresie jednego bajta:

- By wysłać liczbę całkowitą przy pomocy metody *write* należy zastosować przekształcenia na tablice bajtów, np. kod *opakowuje* cyfrę dwa w jednoelementową tablice bajtów. Stąd, wysłanie cyfry dwa z użyciem pakietu *pyserial* sprowadza się do następującej linii kodu

  ```
  port.write(bytearray([2]))
  ```

  Podobnie można wysyłać cyfry w kodzie szesnastkowym

  ```
  port.write(bytearray([0x02]))
  ```

- By odczytać informacje przesłane na port szeregowy jako liczbę całkowitą, tj. jej typ danych to *int*, należy skorzystać z funkcji *ord*, której opis znajduje się tu:

  (https://docs.python.org/2/library/functions.html#ord). Można to zilustrować przykładem

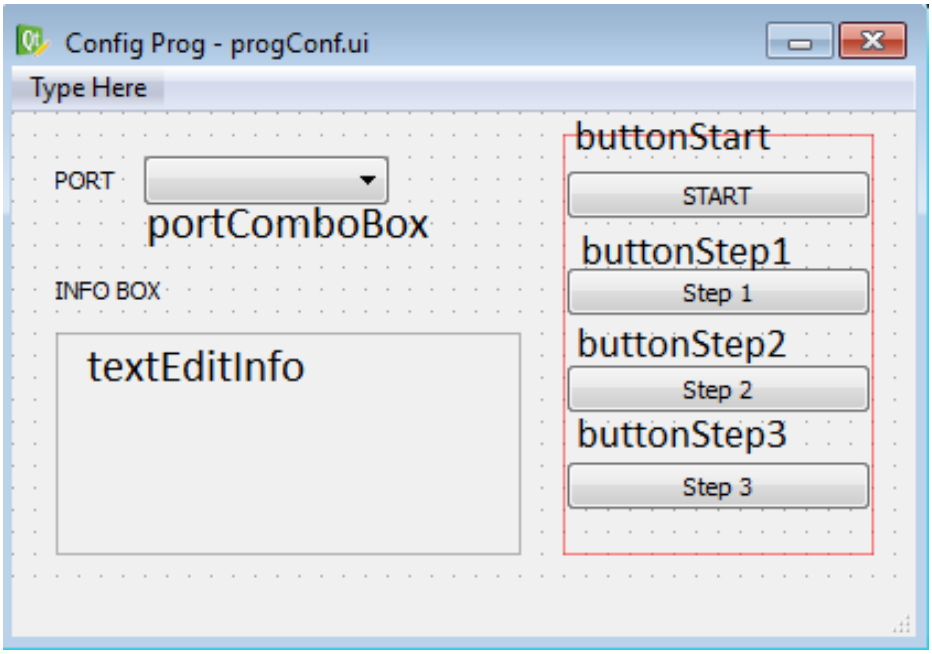
  ```
  x=port.read()
  print ord(x)
  ```

3.2 Zadanie do wykonania

1. W programie *Qt Designer* stworzyć formatkę okna programu. Jego wygląd pokazano na rysunku 1.

Ponadto na rys. 1, naniesiono nazwy, które należy nadać poszczególnym elementom okna. Gdy formatka jest gotowa, należy ją zapisać w folderze KSP.

2. Korzystając z programu *pyuic4* należy przekształcić projekt z Qt Designer-a na kod języka Python. Przykładowe polecenie, które należy wpisać w konsoli języka Python to



Rysunek 1: Formatka okna programu konfiguracyjnego

```
pyuic4 test.ui > test_ui.py
```

3. Z pomocą prowadzącego dokonać analizy wygenerowanego kody w edytorze IDLE.

4. Zapoznać się z działaniem funkcji *enumSerialPorts* dostępnej w pliku *SerialPortUtils* dostarczonego z instrukcją. Można to zrobić wywołując w konsoli polecenie

```
python SerialPortUtils.py
```

5. Skopiować plik *SerialPortUtils* do folderu z plikiem programu w PyQt, i w tym ostatnim poleceniem

```
import SerialPortUtils
```

dołączyć plik *SerialPortUtils*.

6. Korzystając z dokumentacji klasy *QComboBox* (`http://pyqt.sourceforge.net/Docs/PyQt4/qcombobox.html`) zapoznać się z działaniem metody *addItem*. Na podstawie zdobytej wiedzy, i informacji o działaniu funkcji *enumSerialPorts()* należy uzupełnić listę rozwijaną w oknie tworzonego programu nazwami portów. Odpowiedni kod należy dodać do metody *init* w pliku z programem w PyQt. (Wskazówka: użyć pętli *for*)

7. Korzystając z `http://zetcode.com/gui/pyqt4/eventsandsignals/` należy zapoznać się z systemem sygnałów i slotów w PyQt. Następnie należy uzupełnić kod tak, by powstały cztery funkcje, których kod wykona się po wybraniu przycisków dostępnych w oknie programu

8. W funkcji związanej z przyciskiem Start utworzyć obiekt klasy *serial.Serial(portName)* i zapisać go w polu klasy związanej z oknem programu. Jako nazwę portu należy wstawić aktualnie wybrany port w elemencie ComboBox. (wskazówka: sprawdzić działanie metod *current*()* w dokumentacji klasy *QComboBox*)

9. Do funkcji związanych z przyciskami *Step** dodać kod konfigurujący program *SerialDeviceEmulator*

10. Wykorzystując funkcje *addTextToBox* któref definicja to

```
def addTextToBox(self,newText):
        oldText = self.ui.boxTextEdit.toPlainText()
        self.ui.boxTextEdit.setText(newText+"\n"+oldText)
```

dodać do pkt. 9 wyświetlanie komunikatów w sekcji *INFO* okna programu